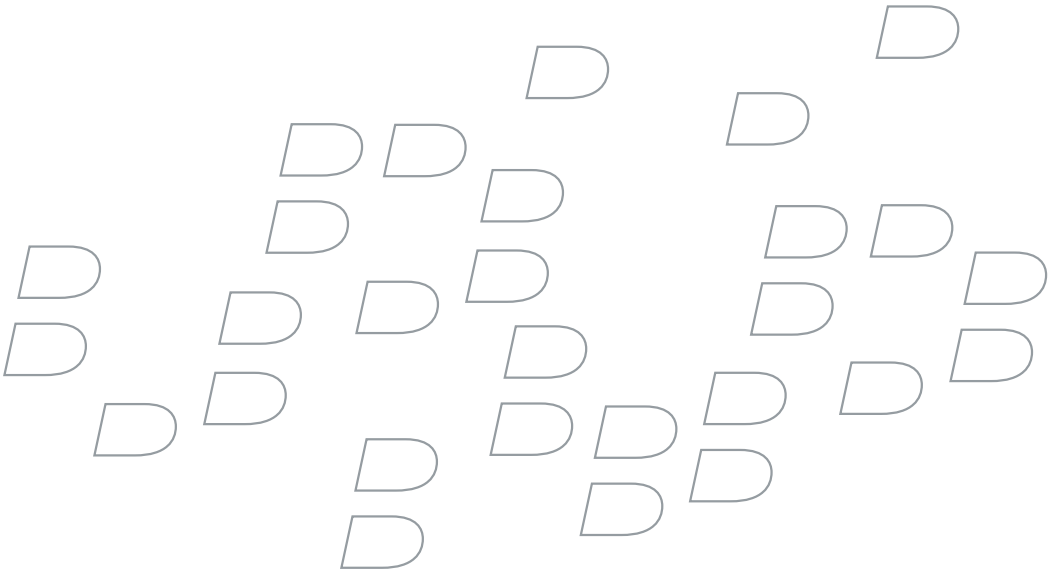

BlackBerry MIDlet Developer Guide

BlackBerry MIDlet Developer Guide

Last modified: 1 February 2006

Part number: SWD_X_JDE(EN)-019.002

At the time of publication, this documentation is based on the BlackBerry Java Development Environment Version 4.1.

Send us your comments on **product documentation:** https://www.blackberry.com/DocsFeedback.

©2006 Research In Motion Limited. All Rights Reserved. The BlackBerry and RIM families of related marks, images, and symbols are the exclusive properties of Research In Motion Limited. RIM, Research In Motion, "Always On, Always Connected", the "envelope in motion" symbol, BlackBerry, and the BlackBerry logo are registered with the U.S. Patent and Trademark Office and may be pending or registered in other countries.

The Bluetooth word mark and logos are owned by the Bluetooth SIG, Inc. and any use of such marks by Research In Motion Limited is under license. Microsoft, Windows, Exchange, and Outlook are registered trademarks of Microsoft Corporation in the United States and/or other countries. Java is a trademark of Sun Microsystems, Inc. in the U.S. and other countries. IBM, Lotus, and Domino are registered trademarks of International Business Machines Corporation in the United States. All other brands, product names, company names, trademarks and service marks are the properties of their respective owners.

The BlackBerry device and/or associated software are protected by copyright, international treaties and various patents, including one or more of the following U.S. patents: 6,278,442; 6,271,605; 6,219,694; 6,075,470; 6,073,318; D445,428; D433,460; D416,256. Other patents are registered or pending in various countries around the world. Visit www.rim.com/patents.shtml for a list of RIM applicable patents.

This document is provided "as is" and Research In Motion Limited and its affiliated companies ("RIM") assume no responsibility for any typographical, technical or other inaccuracies in this document. RIM reserves the right to periodically change information that is contained in this document; however, RIM makes no commitment to provide any such changes, updates, enhancements or other additions to this document to you in a timely manner or at all. RIM MAKES NO REPRESENTATIONS, WARRANTIES, CONDITIONS OR COVENANTS, EITHER EXPRESS OR IMPLIED (INCLUDING WITHOUT LIMITATION, ANY EXPRESS OR IMPLIED WARRANTIES OR CONDITIONS OF FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, MERCHANTABILITY, DURABILITY, TITLE, OR RELATED TO THE PERFORMANCE OR NON-PERFORMANCE OF ANY SOFTWARE REFERENCED HEREIN OR PERFORMANCE OF ANY SERVICES REFERENCED HEREIN). IN CONNECTION WITH YOUR USE OF THIS DOCUMENTATION, NEITHER RIM NOR ITS AFFILIATED COMPANIES AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES OR CONSULTANTS SHALL BE LIABLE TO YOU FOR ANY DAMAGES WHATSOEVER BE THEY DIRECT, ECONOMIC, COMMERCIAL, SPECIAL, CONSEQUENTIAL, INCIDENTAL, EXEMPLARY OR INDIRECT DAMAGES, EVEN IF RIM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, INCLUDING WITHOUT LIMITATION, LOSS OF BUSINESS REVENUE OR EARNINGS, LOST DATA, DAMAGES CAUSED BY DELAYS, LOST PROFITS, OR A FAILURE TO REALIZE EXPECTED SAVINGS.

This document might contain references to third party sources of information, hardware or software, products or services and/or third party web sites (collectively the "Third-Party Information"). RIM does not control, and is not responsible for, any Third-Party Information, including, without limitation the content, accuracy, copyright compliance, compatibility, performance, trustworthiness, legality, decency, links, or any other aspect of Third-Party Information. The inclusion of Third-Party Information in this document does not imply endorsement by RIM of the Third Party Information or the third party in any way. Installation and use of Third Party Information with RIM' s products and services may require one or more patent, trademark or copyright licenses in order to avoid infringement of the intellectual property rights of others. Any dealings with Third Party Information, including, without limitation, compliance with applicable licenses and terms and conditions, are solely between you and the third party. You are solely responsible for determining whether such third party licenses are required and are responsible for acquiring any such licenses relating to Third Party Information. To the extent that such intellectual property licenses may be required, RIM expressly recommends that you do not install or use Third Party Information until all such applicable licenses have been acquired by you or on your behalf. Your use of Third Party Information shall be governed by and subject to you agreeing to the terms of the Third Party Information licenses. Any Third Party Information that is provided with RIM' s products and services is provided " as is" . RIM makes no representation, warranty or guarantee whatsoever in relation to the Third Party Information and RIM assumes no liability whatsoever in relation to the Third Party Information even if RIM has been advised of the possibility of such damages or can anticipate such damages.

Research In Motion Limited
295 Phillip Street
Waterloo, ON N2L 3W8
Canada

Research In Motion UK Limited
Centrum House, 36 Station Road
Egham, Surrey TW20 9LF
United Kingdom

Published in Canada

Contents

1 Creating MIDlets for BlackBerry devices 7
MIDP application overview 7
BlackBerry device Java support 7
BlackBerry Java Development Environment 8
Creating MIDlets 8
Create a MIDlet project 8
Create a MIDlet suite 9
Converting MIDlet .jar files 9
Convert MIDlets using the BlackBerry MDS Connection Service 9
Compile MIDlets using the BlackBerry IDE 10
Compile MIDlets using the Command prompt 10

2 Designing MIDlets 11
Controlling screen display 11
Paint the display screen 11
Buffer UI Images 11
Simulate cellular phone soft keys 11
Using command types 12
Assign functionality to the BlackBerry device Escape button 12
Resource file size limitations 12
MIDlet key mappings 12
MIDlet game action key mappings 12
MIDP function key mappings 13
Trackwheel function key mappings 13

3 Comparing BlackBerry and MIDP UI APIs 15
BlackBerry and MIDP APIs 15
Using screens 15
Layout managers 16
Alerts 16
Using commands 16
Menus 16
Commands 16

Listeners 17
Comparison of BlackBerry and MIDP UI API components 17

4 Using network connections 19
Network protocol support 19
Using TCP socket connections 19
Open a TCP socket connection 19
Open a TCP connection over TLS or SSL 20
Using HTTP connections 20
Open an HTTP connection 20
Open an HTTPS connection over SSL or TLS 21
Using UDP connections 21
Create a UDP connection 22

5 Using application control rules 23
Application control rules 23
CLDC and MIDP API control rules 23
BlackBerry API control rules 25

Index 27

1

Creating MIDlets for BlackBerry devices

MIDP application overview
BlackBerry device Java support
Creating MIDlets
Converting MIDlet .jar files

MIDP application overview

MIDlets are applications that use standard Mobile Information Device Profile (MIDP) version number1.0 or 2.0 Application Program Interfaces (APIs) and Connected Limited Device Configuration (CLDC) APIs only. MIDlets can run on BlackBerry® devices and other devices that support the Java™ Micro Edition (Java ME) platform.

MIDlets can communicate with networks using standard HTTP connections and HTTP over WAP, regardless of the underlying wireless network. On the BlackBerry device, to use a standard HTTP connection (`http://`), the device requires a BlackBerry Enterprise Server™ with the BlackBerry MDS Connection Service component of BlackBerry MDS Services.

MIDlets implement the `javax.microedition.midlet.MIDlet` class and start at the `startApp()` method.

BlackBerry device Java support

BlackBerry devices support the Java ME, Java Technology for the Wireless Industry (JTWI), and BlackBerry API extensions.

BlackBerry devices support the following Java Specification Requests (JSR's):

JSR Number	Description
075	Portable Data Acquisition Package (PDAP) (PIM)
118	MIDP version 2.0
120	Wireless Messaging API (WMA) version 1.1
135	Mobile Media API version 1.1 - BlackBerry devices support the subset defined in MIDP version 2.0.
139	CLDC 1.1
179	Location API - **Note**: The BlackBerry 7520 Wireless Handheld™, BlackBerry 7130e™, and BlackBerry 7100i™ support the Location API (JSR-179).
185	JTWI

Note: The BlackBerry Handheld Software version 3.7 and previous support only MIDP version 1.0. To use applications created with MIDP version 2.0, BlackBerry devices require a version of the BlackBerry Handheld Software greater than 3.7.

BlackBerry Java Development Environment

The BlackBerry Java Development Environment (BlackBerry JDE) includes a set of APIs and tools that you can use to develop custom Java applications for the BlackBerry device:

- BlackBerry Integrated Development Environment (BlackBerry IDE)
- BlackBerry device simulator
- BlackBerry MDS Simulator
- BlackBerry email simulator
- CLDC, MIDP, and BlackBerry APIs
- sample applications
- RAPC command utility

To create, compile, and run MIDlets you require the BlackBerry IDE. The BlackBerry IDE compiles and packages source code into .cod files, which can run on the BlackBerry device. To use the procedures in this guide, users should be familiar with the BlackBerry IDE.

See the *BlackBerry Integrated Development Environment Help* for more information.

Visit www.blackberry.net/developers/ for more information about developing applications for BlackBerry devices.

Creating MIDlets

Create a MIDlet project

1. In the BlackBerry IDE, create or open a workspace.
2. Right-click a project file.
3. Click **Properties**.
4. Click the **General** tab.
5. In the **Title** field, type a title for the MIDlet.
6. In the **Version** field, type a version number.
7. In the **Vendor** field, type the vendor name.
8. Click the **Application** tab.
9. In the **Project type** drop-down list, click **MIDlet**.
10. In the **Name of main MIDlet class** field, type the name of the class in the MIDP application that contains the `startApp()` method.
11. Click **OK**.
12. Compile the project.

The BlackBerry IDE creates a .cod file for the MIDP application. Load the .cod file onto the BlackBerry device. On other MIDP-compatible devices, load the .jar files.

Create a MIDlet suite

A MIDlet suite is a single .jar file that contains one or more MIDlets. (The MIDlets in the suite can share resources and classes.) Use a MIDlet suite to share resources and classes with multiple MIDP applications.

All source files for a MIDlet must be converted into a single .cod file. The source files for a MIDlet suite cannot span multiple projects.

Note: Binary resources cannot exceed 63 KB.

1. In the BlackBerry IDE, create a main project for all source files.
2. In the main project properties in the **Project type** drop-down list, click **MIDlet**.
3. For each MIDlet in the MIDlet suite, create a separate project.
4. In the project properties for each MIDlet project in the suite, in the **Project type** drop-down list, click **Alternate MIDlet Entry Point**. See "Set alternate entry points" on page 9 for more information.

Set alternate entry points

The BlackBerry IDE enables you to define multiple entry points for an application. For example, each MIDlet in a MIDlet suite is a different entry point into the same application. When you build and run the applications, this alternate entry point appears on the BlackBerry device Home screen as a separate application.

1. In the BlackBerry IDE, create a new project.
2. Add an application icon to the project. The project should not contain any other files.
3. Right-click the project file.
4. Click **Properties**.
5. On the **Application** tab, in the **Project type** drop-down list, click **Alternate MIDlet Entry Point**.
6. In the **Alternate entry point for** drop-down list, click the main project for the MIDlet suite.
7. In the **Name of main MIDlet class** field, type the name of the class that extends the `MIDlet` class (for example, `com.company.test.MidletTest`).

   On the BlackBerry device, starting this MIDlet creates a new instance of this MIDlet class.
8. Click **OK**.

Converting MIDlet .jar files

To run MIDlets on a BlackBerry device, the MIDlet must be in the .cod file format.

Convert MIDlets using the BlackBerry MDS Connection Service

The BlackBerry MDS Connection Service feature of the BlackBerry Enterprise Server converts .jar files to .cod files. When a user downloads a MIDlet to the BlackBerry device using the connection service, the connection service converts the .jar file to a .cod file before sending it to the BlackBerry device.

The following versions of the BlackBerry Enterprise Server can convert .jar files to .cod files:

- BlackBerry Enterprise Server version 3.6 or later for Microsoft® Exchange
- BlackBerry Enterprise Server version 2.2 or later for IBM® Lotus® Domino®

Compile MIDlets using the BlackBerry IDE

If you use the BlackBerry IDE to create MIDlets, the BlackBerry IDE creates the necessary files during compilation.

To install an existing MIDlet on a BlackBerry device, the MIDlet must be in the .cod file format.

By default, the BlackBerry MDS Services feature of the BlackBerry Enterprise Server converts .jad files to .cod files.

If you plan to install the MIDlet on BlackBerry devices that do not have a connection to a BlackBerry Enterprise Server, you must convert the MIDlet to the .cod file format. You can convert the MIDlet using the BlackBerry IDE or the BlackBerry RAPC command prompt utility.

1. Create a MIDlet project. See "Create a MIDlet project" on page 8 for more information.
2. Add the MIDlet .jad and .jar files to the project.
3. Right-click the project.
4. Click **Activate Project**.
5. On the **Build** menu, click **Build**.

Note: Include the code for the MIDP application in a single project. You can place resources such as images, in a separate library project.

Compile MIDlets using the Command prompt

The BlackBerry RAPC command prompt compiler converts .java files or a .jar file to a .cod file. The `RAPC` utility is available in the bin directory of your BlackBerry JDE installation directory.

The RAPC command prompt compiler accepts the following parameters:

```
RAPC {import=} {codename=} {-midlet} {jad=} [{filename1.java\filename2.java}
{filename.jar}]
```

Parameter	Description
import	RIM APIs and other application-dependent libraries
codename	code-name for the application; use the same name as the .jar file
-midlet	indicates that the application is a MIDlet
jad	.jad file associated with the application
\<filename1>.java\<filename2>.java	names of the .java files to compile
\<filename>.jar	name of the .jar file

For example, the following command prompt instruction compiles the SampleApp.jar file into a .cod file of the same name.

```
rapc import=net_company_sample.jar;net_company_sample_platform.jar
codename=SampleApp\SampleAppDriver SampleApp\SampleApp.rapc -midlet jad=SampleApp.jad
\Samples\SamplaApp.jar
```

2

Designing MIDlets

Controlling screen display
Using command types
MIDlet key mappings

Controlling screen display

Paint the display screen

Use the `Canvas.paint()` method to clear the entire screen on a BlackBerry device. The `Canvas.paint()` method accepts a single parameter: a `Graphics` object with a clip region that defines the area of the screen that is considered to be obscured. Your implementation of the `paint` method must redraw every pixel within the invalid region. If not, an application might display incorrectly within this region, and a `repaint()` request from an external source, such as the BlackBerry application framework, will invalidate the screen contents.

Note: MIDlets support .png files with bit depths of 1, 2, 4, 8, 16, 24 and 32 bits.

Buffer UI Images

Although MIDP applications require double buffering, the BlackBerry user interface (UI) implementation is inherently double-buffered. Therefore UI images do not require double buffering. Essentially, the drawing functions in the `net.rim.device.api.ui.Graphics` objects are drawn to an off-screen bitmap. When you repaint the display screen, the system draws this off-screen bitmap instead of sequentially painting the screen.

The fields and field managers in the Blackberry UI use this drawing paradigm.

Simulate cellular phone soft keys

The BlackBerry device displays commands normally associated with cellular phone soft keys as menu items on the application menu.

When the menu is active, `paint()` events do not update MIDlet screens. The application might appear paused, even though the background threads update the internal states.

To solve this problem, place the code that updates the application in the `paint()` method, and invoke `repaint()` in a continuous loop from the main application thread.

When the menu is closed, `repaint()` invokes `paint()` to update the screen and the application state. When the menu is open, `paint()` is not invoked, and the application appears paused.

Using command types

The BlackBerry application framework uses command types to prevent duplicate menu items. Use the appropriate command types for your commands to make sure that menu items on a menu are unique.

For example, when a command closes a MIDlet, to prevent the BlackBerry platform from automatically adding a **Close** item to the menu, verify that the MIDlet contains a command of type `Command.EXIT`. If a MIDlet does not use the correct command type, the menu might contain both an **Exit** and a **Close** item.

Note: When a MIDlet adds Command objects to a displayable object, they are displayed in a BlackBerry-style menu.

Assign functionality to the BlackBerry device Escape button

You can assign the BlackBerry device **Escape** button to one of the following commands: `Command.CANCEL`, `Command.BACK`, or `Command.EXIT`.

To provide a response when the user presses the **Escape** button, your MIDlet should implement one of these commands. If the MIDlet implements more than one command, the Escape button uses the commands in the following order:

1. `Command.CANCEL`
2. `Command.BACK`
3. `Command.EXIT`

Resource file size limitations

When creating a MIDlet, the maximum size for each resource file (for example, a .png file) that you can include with MIDlet source code is 63 KB.

MIDlet key mappings

MIDlet game action key mappings

For MIDlet games, action key mappings depend on the BlackBerry device type:

Device	Action	Key
BlackBerry 7290 Wireless Handheld™, BlackBerry 7100 Series, and BlackBerry 8700 Series of wireless handhelds	up	2
	left	4
	down	8
	right	6
BlackBerry 7250 Wireless Handheld™, and BlackBerry 7270 Wireless Handheld™	up	R
	left	D
	down	G
	right	C

MIDP function key mappings

BlackBerry keys mappings relate to various MIDP functions (the mappings are not case sensitive):

Key	MIDP <function>
Space	`Canvas.FIRE`
Q	`Canvas.GAME_A`
W	`Canvas.GAME_B`
O	`Canvas.GAME_C`
P	`Canvas.GAME_D`
R, U	`Canvas.UP`
F, J, C, N	`Canvas.DOWN`
D, H ,A ,L	`Canvas.LEFT`
G, K, S	`Canvas.RIGHT`

Trackwheel function key mappings

The BlackBerry trackwheel mapping relates to various MIDP functions:

Action	Procedure	MIDP methods
Scroll down.	Roll the trackwheel clockwise.	Invokes the `keyPressed()` and `keyReleased()` methods of the current `Canvas` object; each invoked with the `keyCode` parameter set to `Canvas.DOWN`.
Scroll right.	Press the **Alt** key and roll the trackwheel clockwise.	Invokes the `keyPressed()` and `keyReleased()` methods of the current `Canvas` object with the `keyCode` parameter set to `Canvas.RIGHT`.
Scroll up.	Roll the trackwheel counter-clockwise.	Invokes the `keyPressed()`and `keyReleased()` methods of the current `Canvas` object; each invoked with the `keyCode` parameter set to `Canvas.UP`.
Scroll left.	Press the **Alt** key and roll the trackwheel counter-clockwise.	Invokes the `keyPressed()` and `keyReleased()` methods of the current `Canvas` object with the `keyCode` parameter set to `Canvas.LEFT`.

3

Comparing BlackBerry and MIDP UI APIs

BlackBerry and MIDP APIs
Comparison of BlackBerry and MIDP UI API components

BlackBerry and MIDP APIs

When you write applications for BlackBerry devices, you can use one of two user interface API's:

- MIDP UI APIs (the `javax.microedition.lcdui` package)
- BlackBerry UI APIs (the `net.rim.device.api.ui` packages)

If you are writing an application for an MIDP-compliant device, use the MIDP UI APIs. If you are writing an application specifically for BlackBerry devices, use the BlackBerry UI APIs. BlackBerry UI APIs are designed to provide access to specific features of the BlackBerry device and enable more sophisticated UI layout and interaction.

Note: The BlackBerry UI APIs and MIDP UI APIs are mutually exclusive. You cannot use both API sets in the same application.

See the *BlackBerry Application Developer Guide Volume 1: Fundamentals* for more information about using the BlackBerry UI APIs.

Using screens

The BlackBerry UI API structures components differently than the MIDP UI API. With the BlackBerry UI API, you can combine any field object with other field objects.

Note: The BlackBerry UI considers certain `Displayable` objects in MIDP such as `List` and `Alert` as field components.

You can also combine components regardless of level, and the combinations of components you can use for each screen are unlimited. Combine any number of field objects together into a `Manager`, such as a `ListField`, `GaugeField`, or numerous `BitmapField` objects. In this respect, the BlackBerry UI API is more extensible than the MIDP UI API, which enables you to create complex application interfaces more efficiently.

The MIDP UI API separates UI components into high, middle, and low-level APIs. Since you can only aggregate MIDP items to `Form` displayables, you cannot combine `List` and `Alert` objects with other components in the MIDP UI API.

Using the MIDP UI API components, you cannot display more than one `Displayable` object on the screen at a time. For example, you cannot combine a `List` with another type of UI component, such as a `Gauge`.

Layout managers

The BlackBerry UI API provides layout managers that enable you to display fields vertically, horizontally, or both. You declare a layout manager at the top level of your screen, and then add field components to it.

The closest relation to a layout manager in the MIDP UI API is the `Form` component. It aggregates `Item` objects vertically on the screen.

When deciding if you should use the BlackBerry UI APIs or the MIDP UI APIs, consider how you intend to organize content on the screen. If your interface involves frequent scrolling between fields or a horizontal field layout, you might want to use the layout managers of the BlackBerry UI API.

Alerts

The BlackBerry UI API has two main types of `Screen` objects: `PopupScreen`, which contains the `Dialog` and `Status` subclasses, and `FullScreen`, which contains the `MainScreen` subclass. A `PopupScreen` is not treated as a specific `Screen` object; you are not required to push it onto the context stack.

The BlackBerry UI API enables you to display supplementary application information in a pop-up screen without interrupting the "flow" of the application. Users can view the pop-up screen without leaving the main screen.

The MIDP UI API contains an `Alert` object. Since the MIDP UI API is intended for devices with minimal displays, such as mobile phones, an `Alert` object functions differently from the `PopupScreen` in the BlackBerry UI API. For example, `Alert` objects are screens that use the entire display.

Using commands

BlackBerry devices display commands as menu items. MIDP devices typically display commands as buttons on the screen.

Menus

The MIDP UI API prioritizes commands according to a device's specifications and assigns certain commands to screen buttons. BlackBerry devices organize commands on a menu that is accessible through the trackwheel. You can specify both the order of menu items and the menu item that is selected by default when the menu opens.

Commands

Use the BlackBerry UI API to add commands to the application's `Menu` object. Menu items can be applied to an entire application or to a particular application screen. You can tailor specific menu items to specific parts of your application by creating context menu items.

Use MIDP UI API components to assign commands directly to `Displayable` objects.

Listeners

The BlackBerry UI API is designed to offer a more flexible implementation of listeners than the MIDP UI API, based on a more involved level of components that can be structured on a screen. The BlackBerry UI API contains a generic event listener model, which enables you to implement listeners for many types of events, such as a user pressing a key or clicking the trackwheel, and field focus and scrolling changes.

The MIDP UI API includes one default type of listener, `CommandListener`. Register one `CommandListener` for each `Displayable` object. After you register a `CommandListener` with a `Displayable` object, write code that performs an action when the `Displayable` object is selected. For example, assign a command listener to a `Form`, and create code that performs an action when the user selects an item on the `Form`. The MIDP UI API also contains basic keyboard input event handling functions.

Comparison of BlackBerry and MIDP UI API components

BlackBerry	MIDP	Comments
`ActiveAutoTextEditField`	–	`ActiveAutoTextEditField` is an extension of `AutoTextEditField` that finds and highlights active text in a text string.
`ActiveFieldContext`	–	`ActiveFieldContext`does not have a MIDP equivalent.
`ActiveRichTextField`	–	`ActiveRichTextField` is an extension of `RichTextField`.
`AutoTextEditField`	–	`AutoTextEngine` is specific to the BlackBerry device.
`BasicEditField`	`TextField`	`TextField` is an editable text field.
`BitmapField`	`Image`	The BlackBerry UI `BitmapField` objects are similar to the `Image` object in the MIDP UI. `Bitmap` is equivalent to `ImageItem`.
`ButtonField`	–	Create a `ButtonField` equivalent in MIDP by using a variety of drawing and focusing methods in a `Canvas` object.
`CheckboxField`	`ChoiceGroup`	The MIDP equivalent of a `Checkboxfield` is a `ChoiceGroup` with multiple selection specified.
`ChoiceField`	–	The MIDP `ChoiceGroup` item is similar to a `RadioButtonField` or `CheckboxField`.
`DateField`	`DateField`	`DateField` objects are editable in both the BlackBerry and MIDP APIs.
`Dialog`	–	MIDP does not support pop-up screen functionality. The `Alert` object serves as the MIDP pop-up screen component.
`EditField`	–	`EditField` is an extension of `BasicEditField` that enables users to type special characters in the field.
`GaugeField`	`Gauge`	A MIDP `Gauge` must be declared as interactive or non-interactive. For a `GaugeField` to be interactive, it must be created with the `Field.EDITABLE` style.
`LabelField`	`StringItem`	A non-editable text field.
`ListField`	`List (Displayable)`	A `List` in MIDP provides a list of options, which can be used instead of a menu. Since `List` is a `displayable` object, you cannot add other UI components to the screen at the same time.
`Menu`	–	Commands in a MIDP application appear as menu items on the BlackBerry device.
`NullField`	–	A `NullField` is a field that has no size. It is often used as a source for a focus change listener.

BlackBerry	MIDP	Comments
NumericChoiceField	–	–
ObjectChoiceField	–	–
ObjectListField	–	–
PasswordEditField	TextField	In MIDP, a PASSWORD constraint constant can be added to a TextField; characters are not displayed on the screen.
RadioButtonField	ChoiceField	The MIDP equivalent of a RadioButtonField is a ChoiceField with an explicit selection specified.
RadioButtonGroup	–	Options in a MIDP ChoiceField are grouped by default.
RichTextField	–	In MIDP, style attributes (such as STYLE_ITALIC and STYLE_BOLD) can be applied to text drawn on a Canvas object.
SeparatorField	–	The separator is a user interface component that is specific to standard BlackBerry application interfaces.
Status	Alert	An Alert can be modal requiring user interaction, or timed closing automatically. An Alert is not a pop-up screen, but it serves a similar purpose.
–	Ticker	The BlackBerry UI API does not contain an equivalent item for the MIDP Ticker item.
TreeField	–	–

4

Using network connections

Network protocol support
Using HTTP connections

Network protocol support

The BlackBerry device provides a Java ME platform that is designed to support the following network protocols:

- Transmission Control Protocol sockets (TCP sockets)
- TCP Secure Sockets Layer (SSL) or Transport Layer Security (TLS)
- Hypertext Transfer Protocol (HTTP)
- HTTP over Secure Socket Layer (HTTPS) or TLS
- User Datagram Protocol datagrams (UDP)

MIDP version1.0 specifies HTTP as the standard networking protocol. Although HTTP is useful for most data exchanges, many applications require networking capabilities outside the standard request and response models of most browsers.

Using TCP socket connections

The MIDP Generic Connection Framework (GCF) supports TCP socket connections.

Note: The BlackBerry Handheld Software version 3.6.1 or later supports the `ServerSocketConnection` and socket options such as `DELAY`, `LINGER`, `KEEPALIVE`, `RCVBUF`, and `SNDBUF`.

Open a TCP socket connection

Use `Connector.open()` and specify the socket protocol. (Cast the returned object as a `StreamConnection`.)

```
StreamConnection conn;
conn = (StreamConnection)Connector.open("socket://<host>:<port>;deviceside=true");
```

To open a direct TCP connection from the BlackBerry device, set the `deviceside` parameter to `true`.

```
conn = (StreamConnection)Connector.open("socket://testserver:600;deviceside=true");
```

The `deviceside` parameter is optional and determines whether the socket connection is in proxy or direct mode.

Value	Description
true	The BlackBerry device uses a direct TCP connection. If the BlackBerry device does not support direct TCP, the connection fails.
false	The BlackBerry device uses a proxy TCP connection through the BlackBerry Enterprise Server. If the BlackBerry device is not registered with a BlackBerry Enterprise Server, the connection fails.
not specified or none	The BlackBerry 6510 Wireless Handheld™ uses a direct TCP connection. Other BlackBerry devices use a proxy TCP connection by default.

Note: In most cases, omit the `device-side` parameter to use a direct TCP connection on the BlackBerry 6510™ and a proxy TCP connection on BlackBerry devices that do not support direct TCP.

Open a TCP connection over TLS or SSL

Use `Connector.open()` using the `tls://` or `ssl://` protocol:

```
Connector.open("ssl://<host>:<port>;deviceside=true"); //use SSL
Connector.open("tls://<host>:<port>;deviceside=true"); //use TLS
```

As with standard TCP connections, you can specify the optional `deviceside` parameter. If you do not include the `deviceside` parameter or if you specify `deviceside=false`, you can set one of the following parameters for the secure layer:

Parameter	Description
`END_TO_END_REQUIRED`	Uses an end-to-end TLS or SSL connection from the BlackBerry device to the target web server; if an end-to-end TLS or SSL connection cannot be set, close the connection. For example: `Connector.open("tls://testserver:600;END_TO_END_REQUIRED");`
`END_TO_END_DESIRED`	Uses an end-to-end TLS or SSL connection from the BlackBerry device to the target web server if the BlackBerry device supports secure connections; if the BlackBerry device does not support end-to-end TLS, and the user permits proxy TLS connections, the device uses a proxy connection.
not specified	Uses the BlackBerry Enterprise Server as a proxy for the secure connection: By default, proxy TCP connections use proxy TLS.

Note: Direct TCP connections always use end-to-end TLS.

Using HTTP connections

Since HTTP connections are carried over TCP, the same parameters that apply to TCP socket-layer connections apply to HTTP connections.

Open an HTTP connection

1. Create an `HTTPConnection` object.

```
HttpConnection conn = null;
```

2. Invoke the `Connector.open()` method and cast this object as an `HTTPConnection`.
```
conn = (HttpConnection)Connector.open("http://testserver/index.htm");
```

Note: Since HTTP is carried over TCP, the same rules regarding the `deviceside` parameter that apply to TCP connections apply to HTTP connections.

3. Use `HttpConnection` class methods to set the HTTP request method and other HTTP headers.
```
conn.setRequestMethod(HttpConnection.POST);
conn.setRequestProperty("If-Modified-Since", "12 May 2005 19:43:31 GMT");
conn.setRequestProperty("User-Agent", "BlackBerry/3.8");
conn.setRequestProperty("Content-Language", "en-US");
```
4. To send and receive data, acquire input and output streams using the `openInputStream()` and `openOutputStream()` methods.
```
InputStream in = conn.openInputStream();
```

Open an HTTPS connection over SSL or TLS

1. Complete the "Open an HTTP connection" procedure.
2. Set the protocol to HTTPS.
```
HttpConnection conn;
conn = (HttpConnection)Connector.open("https://host:443/");
```

For HTTPS connections, the BlackBerry device uses TLS by default. If TLS does not work, the BlackBerry device switches to SSL automatically.

Note: Since HTTP is carried over TCP, the same rules regarding the `END_TO_END_REQUIRED` and `END_TO_END_DESIRED` parameters for TCP connections apply to HTTP connections. For example, the following code requests an HTTP connection over end-to-end TLS: `Connector.open("https://host:443/;END_TO_END_DESIRED");`

Using UDP connections

The BlackBerry device supports datagram connections using the User Datagram Protocol (UDP). UDP is designed to enable your application to communicate with standard network services. Datagrams are independent packets of data that your application can send over the network. Unlike HTTP connections, datagram connections are stateless: packets can arrive in any order, and delivery is not guaranteed.

Your application must format the data payload of request datagrams so that the data payload conforms to the standards of the network service your application is communicating with. Your application must also be able to parse the datagrams that the server sends to it.

To use a UDP connection, you must have access to an infrastructure that supports connections to a wireless network, including an access point name (APN) for GPRS networks.

Note: Since datagram connections do not use the BlackBerry network infrastructure, communication is not encrypted.

Use the `javax.microedition.io.DatagramConnection` interface, which extends the `Connection`, to define the connections that send and receive datagrams. Implement the `Datagram` interface to define packets that are sent and received over a datagram connection.

Create a UDP connection

1. Open a UDP connection by invoking the `Connector.open()` method using the following format, specifying UDP as the protocol:

   ```
   (DatagramConnection)Connector.open("udp://host:dest_port[;src_port]/apn");
   ```

Parameter	Description
`host`	destination host address to which to send data, in dotted decimal format, such as 112.11.11.11
`dest_port`	destination port number at the host address to which to send data
`src_port`	source port number on the BlackBerry device on which to receive incoming data
`apn`	The APN to use to connect to the network, in string format (for GPRS networks only)
`/rport=src_port`	port number for listening; for use with the device simulator

2. Retrieve a `DatagramConnection` object by invoking the `Connector.open()` method, specifying UDP as the protocol:

   ```
   private static String address = "udp://121.0.0.0:2332;6343/test_apn";
   DatagramConnection conn = null;
   conn = (DatagramConnection)Connector.open(address);
   ```

3. Perform any of the following actions:.

Action	Procedure
Send data to a specific location.	Specify a destination host and port number.
Receive data on a BlackBerry device.	Specify a source port number.

For incoming-only connections, you can omit the destination port, in which case the connection can receive data from all ports at the specified host.

Note: To open a connection on a non-GPRS network, do not specify the APN. You must still include the front slash after the source port value. For example, the address for a CDMA network connection would be `udp://121.0.0.0:2332;6343/`.

5

Using application control rules

Application control rules
CLDC and MIDP API control rules
BlackBerry API control rules

Application control rules

BlackBerry Application control rules are designed to provide administrators with the ability to control the functions and data stores that an application on a BlackBerry device can access. Administrators can specify rules for an application or class of applications. For example, administrators can use application control to make sure that a game on a BlackBerry device cannot access the phone API. If a method attempts to access an API element that it does not have access to, the method throws a `ControlledAccessException`. Use the IT policy rules in this guide to ensure your applications handle exceptions when they access an API element they do not have access to.

Notes: Application control rules apply to third party applications only.

MIDP and CLDC APIs are available to a MIDlet on any Java ME device. BlackBerry APIs are available to applications on BlackBerry devices only. When a MIDP application accesses a BlackBerry API, the MIDlet becomes a BlackBerry-specific MIDlet and will only work on the BlackBerry platform.

CLDC and MIDP API control rules

Applications that only use the CLDC and MIDP APIs are subject to the following CLDC and MIDP rules:

IT policy rule	Values	Description	Relevant API elements	Default value
Bluetooth Serial Profile	Allowed Not permitted	Controls the ability to access the Bluetooth® serial profile API. The Bluetooth serial profile API enables a third-party application to use functionality similar to serial ports over Bluetooth. If the IT policy is not set, access to the Bluetooth serial profile API is enabled by default.	`Bluetooth API` `Connector.open()`	Allowed

IT policy rule	Values	Description	Relevant API elements	Default value
Device GPS	Allowed Prompt User Deny Not permitted	Controls the ability to access the GPS APIs on a BlackBerry device. By default, if a device has an IT Policy that does not contain application control data, GPS API access is turned on and prompting is turned off for all third-party applications. If the device has an IT Policy that contains application control data, the default action is to prompt the user for permission. If no IT Policy is set on the device, the user is not prompted.	`Location class`	Allowed or Prompt User
External Network connections	Allowed Prompt User Not permitted	Controls the ability to send and receive external connections. External connections cannot retrieve information from inside a firewall. Direct TCP and WAP are examples of external connections.	`Connector.open()`	Prompt User
Internal Network connections	Allowed Prompt Not permitted	Controls the ability to send and receive connections sent through a corporate network using the BlackBerry MDS Services.	`Connector.open()`	Prompt User

Phone Access	Allowed
	Prompt
	Not permitted
		Controls whether the application can initiate phone calls. The phone API (`net.rim.blackberry.api.phone`) and the invoke API (`net.rim.blackberry.api.invoke`) provide unique capabilities, such as invoking the phone application on or retrieving the phone logs from the device.
		By default, phone calls are enabled and prompting is turned on for all third-party applications.
			`DirectConnect`
			`Invoke.invokeApplication()`
			`PhoneLogs class`
			`Phone class`
				Prompt User

PIM Data Access	Allowed Not permitted	Controls the ability to access the PIM API. The PIM API (`javax.microedition.pim`) provides applications with access to personal information management (PIM) data, such as contacts, tasks, and events.	`ApplicationMenuItem class` `PIM API`	Allowed

Note: See the *BlackBerry Enterprise Server for Microsoft Exchange Version 4.0 Administration Guide* for information about how a BlackBerry Enterprise Server administrator sets application control rules.

See the *BlackBerry Wireless Handheld User Guide* for your device for information about how to set rules on a BlackBerry device.

BlackBerry API control rules

IT policy rule	Values	Description	Relevant API elements	Default value
BlackBerry device Keystore	Allowed Not permitted	Controls whether the application can access the device key store API. The device key store stores the users public certificates and private keys. If an IT policy is not set, access to the device key store API is enabled by default.	`KeyStore interface`	Allowed
BlackBerry device Keystore Medium Security	Allowed Not permitted	Controls whether a third party application can use the key store medium security level when accessing private keys stored in any key store. When a private key is inserted into a key store, the security level is set to low, medium, or high. If the security level is set to low the user is never prompted for a key store password when accessing that key. If the security level is medium, the key store caches the user's key store password when the user first types it; and the keystore does not prompt the user again until the memory cleaner clears the cached password. If the security level is high, the key store prompts the user for a password (or if the user typed the password recently, the keystore prompts the user to confirm access to the private key). If key store medium security is not enabled, the key store forces private key access for medium security to high security. If an IT policy is not set, key store medium security is enabled by default.	`KeyStore interface`	Allowed
Browser Filters	Allowed Not permitted	Controls whether the application can register a browser filter with the browser. A browser filter is an application that resides in between the data received by the browser and the UI displayed to the user. It enables an application to add, update, modify, or remove data displayed to the user.	`HttpFilterRegistry class`	Not permitted
Event Injection	Allowed Not permitted	Controls whether the application can inject events into the system. The Event Injector API enables a third party application to inject keypress and trackwheel events. With this feature, an application can simulate events that did not actually occur on the device.	`EventInjector class`	Not permitted

IT policy rule	Values	Description	Relevant API elements	Default value

Inter Process Communication	Allowed
	Not permitted
		Controls whether the application can access to methods or APIs that provide an application with the ability to share information with another process on the device.	`Application.addGlobalEventListener()`
			`RuntimeStore class`
			`PersistentStore class`
			`StringPatternRepository class`
			`File class`
			`Clipboard class`
				Allowed

Local Network Connections	Allowed
	Not permitted
		Controls the ability to make local connections to the device similar to the Internal and External Connections allowed features. USB and serial connections are examples of local connections.	`Connector.open()`
			`USBPort class`
			`SerialPort class`
			`Application.addIOPortListener()`
				Allowed

Message Access	Allowed Not permitted	Controls access to the mail API. The mail API (`net.rim.blackberry.api.mail`) enables applications to access mail stored on the device.	`Session class` `ApplicationMenuItem class`	Allowed

Index

A

alerts, 16
APIs, 15
application, 23
application control permissions, 23

C

commands, 12, 16
comparing MIDP and BlackBerry APIs
  commands, 16
  screens, 15
  UI components, 17
connections
  HTTP, 20
  network protocol support, 19
  secure HTTPS, 21
  TCP socket, 19
  UDP, 21
creating MIDlets
  MIDlet suites, 9
  projects, 8

D

deploying MIDlets
  BlackBerry IDE, 10
  BlackBerry MDS Connection Service, 9
designing MIDlets
  buffer UI images, 11
  key mappings, 12
  screen display, 11
  screen painting, 11
  using command types, 12
dialogs, 16

I

IT policies, 23

J

jad files, 9
jar files, 9
Java support, 7

L

Listeners, 17

M

menus
  compared to soft keys, 11
MIDlet suites, 9
MIDlets
  application control permissions, 23
  commands, 12
  creating, 8
  deploying, 9
  game actions keys, 12
MIDP applications
  See also MIDlets

P

permissions for applications, 23
PIM
  personal information manager, 23
pop-up screens, 15

S

secure connections, 21
soft keys, 11, 16
SSL
  see Secure Sockets Layers, 21

T

TCP connections, 19
TLS
  see Transport Layer Security, 21

trackwheel functions, 13

U

UDP connections, 21
User interface APIs, 11

W

WAP
    Wireless Access Protocol, 24

©2006 Research In Motion Limited
Published in Canada.